VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Internetová aplikace evidující tipy sázkařů

Internet Application Administrating Bettor's Picks

2012 Lukáš Grygar

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Lukáš Grygar**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Internetová aplikace evidující tipy sázkařů
Internet Application Administrating Bettor's Picks

Zásady pro vypracování:

Cílem práce je vytvoření moderní webové aplikace, která umožní sportovním sázkařům evidenci jejich tipů a vyhodnocování úspěšnosti tipů. Zadání lze shrnou t v těchto bodech:
1. Prozkoumejte existující nástroje dotýkající se oblasti evidence tipů při sázení. Analyzujte, jaké údaje je vhodné pro sportovní sázkaře evidovat a jaké výstupy jsou užitečné pro budoucí sázení.
2. Navrhněte a realizujte aplikaci usnadňující evidenci tipů jednotlivých sázkařů. Aplikace by měla umožnit evidovat různé druhy sázek (například na vítěze, nebo na počet bodů) v různých kategoriích a evidovat jejich úspěšnost.
3. Na základě vložených údajů bude aplikace schopna vyhodnocovat různé přehledové údaje o sázení a vhodně (například například pomocí grafů) je zobrazovat. Tyto údaje by měli usnadnit sázejícím další tipování.
Při implementaci využijte internetového vývojového rámce Django. K vlastní implementaci použijte programovací jazyk Python.

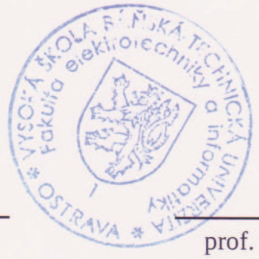
Seznam doporučené odborné literatury:

Pilgrim, M. 2004 Dive into Python. APress.
Domovská stránka projektu Django: http://www.djangoproject.com/

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Marek Běhálek, Ph.D.**

Datum zadání: 19.11.2010
Datum odevzdání: 04.05.2012



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 4. května 2012

Lukáš Grygar

.....

Rád bych na tomto místě poděkoval vedoucímu práce Ing. Markovi Běhálkovi, Ph.D. za jeho podněty a připomínky. A dále mé rodině za morální podporu během studia.

Abstrakt

Tato bakalářská práce seznamuje čtenáře s problematikou vedení evidence sázkařských tipů při kurzovém sázení. Popisuje návrh a realizaci webové aplikace pomocí webového frameworku Django, která sportovním sázkařů umožní získat přehled o jejich dosavadních výsledcích a usnadní jim další tipování. Dále prakticky ukazuje nasazení na cloudové platformě a rozebírá také možnosti mozetizace.

Klíčová slova: Python, Django

Abstract

This bachelor thesis introduces readers to the problems of keeping records for sports betting. Describes the design and implementation of web application using Django web framework that allows sports bettors to get an overview of their current results and facilitate their sports betting in future. Shows deployment on the cloud platform and discusses the possibilities of monetization.

Keywords: Python, Django

Seznam použitých zkratek a symbolů

CAPTCHA	–	Completely automated public Turing test to tell computers and humans apart
CRUD	–	Create, read, update and delete
CSS	–	Cascading Style Sheets
ERM	–	Entity-relationship Model
HTML	–	HyperText Markup Language
HTTP	–	HyperText Transfer Protocol
ORM	–	Object-relational Mapper
PPC	–	Pay Per Click
SQL	–	Structured Query Language
URL	–	Uniform Resource Locator
WSGI	–	Web Server Gateway Interface

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Sportovní sázkaři se dříve či později mohou setkat s problémem evidence jejich tipů.

Potřebu evidence svých tipů pociťují především zkušenější sázkaři, jimž tato evidence může pomoci k odhalení vlastních chyb a nedostatků a toto sebepoznání jim do budoucna může přinést zlepšení jejich úspěšnosti.

Tito sázkaři si pak začnou vést evidenci obvykle v různých tabulkových procesorech jako je například velmi rozšířený Microsoft Excel. Toto řešení ovšem není vhodné z dlouhodobého hlediska, protože s rostoucími daty se toto řešení stane nepřehledné, ať už z důvodu počáteční nezkušenosti autora nebo z důvodu nerelačních dat. Proto bude pro tyto sázkaře lepší využít již existující řešení.

Vytvoření takovéhoto řešení si za cíl klade tato práce.

Kapitola 2 popisuje již existující řešení v této oblasti, popisuje práci s těmito aplikacemi a jejich funkčnost, kterou lze vypozorovat jejich užíváním.

Kapitola 3 specifikuje požadavky, které jsou kladeny na vyvíjený software.

Kapitola 4 stručně seznamuje s technologiemi využitými pro tvorbu aplikace.

Kapitola 5 analyzuje funkční požadavky, zobrazuje je pomocí diagramů případů užití, dále pak analyzuje vstupní a výstupní data aplikace.

Kapitola 6 popisuje implementaci především pomocí frameworku Django, seznamuje čtenáře s konvencemi tohoto frameworku a s funkcemi, které nabízí. Dále také popisuje integraci software třetích stran, který byl využit v některých částech aplikace.

Kapitola 7 popisuje testování aplikace malou skupinou testerů a částečně diskutuje jejich zpětnou vazbu.

Kapitola 8 popisuje technologie sloužící k produkční nasazení a prakticky demonstruje nasazení.

Kapitola 9 se zamýšlí nad způsoby monetizace takovéhoto typu aplikace.

Kapitola 10 obsahuje zhodnocení dosažených výsledků, osobní přínos a nastiňuje možný budoucí vývoj.

2 Prozkoumání již existujících řešení

Na internetu existuje mnoho portálů, které se věnují sportovnímu sázení. Mnoho z nich umožňuje svým registrovaným členům publikovat své sázkařské tipy s analýzami.

Ovšem tyto portály tuto službu poskytují spíše z důvodu zvýšení své vlastní návštěvnosti a pro sázkaře jako takového nemají zpětnou vazbu, protože neumožňují efektivní vyhodnocování vkládaných dat.

Jako zástupce sofistikované aplikace, která umožňuje vyhodnocování vkládaných dat, jsem vybral BetMagnet.com a zhodnotil principy jeho užívání a funkce.

Jako zástupce spíše komunitní větve jsem vybral BetBook.com, který ovšem žádnou sofistikovanou funkčnost nenabízí.

2.1 BetMagnet.com

V BetMagnetu [19] je po registraci nutné nastavit základní měnu. Dle vyjádření tedy bude u každé sázky uložena vždy hodnota v základní měně a pokud se bude měna sázky lišit od základní měny, tak bude samozřejmě uložena i měna dané sázky. Celkový zisk se vždy vypočte ze základní měny. Zisky či ztráty u sázek, u kterých je užita jiná než základní měna, jsou přepočteny do základní měny dle aktuálního kurzu. Tento přepočet však může být nepřesný, protože neumožňuje přesně evidovat finance u sázkařů, kteří mají účty ve více než jedné měně. Může totiž nastat situace, kdy sázkař má účet veden v dolarech a peníze ze sázkařského účtu vybere až po určité době, která se liší od doby uzavření jednotlivých sázek, výběr tedy proběhne, ale převodní kurz se liší od těch, které převedl BetMagnet. Tato situace je nevýhodná, pokud potřebujeme vědět přesný zisk pouze v jedné pevně stanovené měně. V opačném je tento postup ukládání správný. Pokud bychom ovšem chtěli přesto přesný zisk pouze v určité měně, je nutné oprostit evidenci sázkových tipů od zisků, brát výběr z konta sázkové kanceláře jako jednu jednotku a evidovat tyto zisky spíše pomocí účetního softwaru. BetMagnet dále varuje, že nastavení základní měny je trvalé a v budoucnu nebude možné ho změnit.

Po povinném nastavení základní měny BetMagnet umožňuje další formy personalizace účtu.

První je "Moje sázkovky", což je seznam sázkových kanceláří uživatele. Systém vždy eviduje sázkovou kancelář a měnu, ve které má uživatel vedený účet. Uživatel si může vybrat, kterou sázkovou kancelář si přidá se seznamu sázkových kanceláří, který poskytuje BetMagnet, seznam neposkytuje všechny sázkové kanceláře, a tak je možné přidat si vlastní sázkovou kancelář.

Dále umožňuje spravovat "Moje sporty", což je seznam sportů, na které uživatel sází. Stejně jako u sázkových kanceláří i zde je možné si vybrat z nabídky mnoha sportů a nebo si vytvořit vlastní, pokud ho nabídka neobsahuje.

Umožňuje také spravovat "Moje zdroje", což je zdroj inspirace k podání tipu, udává se název a popis zdroje. Přednastavený zdroj, který je přidán do zdrojů každého uživatele, je vlastně on sám, ale lze také přidávat další zdroje.

Po personalizaci výše zmíněných nastavení již můžeme přejít k samotnému jádru aplikace a tou je přidávání sázkových tipů. Pro přidání tipu je nutno vyplnit datum

podání tipu, které je již přednastaveno, ale je možné jej editovat. Myslím, že není vhodné, aby byla tato volba editovatelná, ale BetMagnet to tak má nastavené. Dále údaje týkající se příležitosti samotné, tedy zdroj, začátek, sport, soutěž, zápas a tip je v jednom poli a posledním údajem je kurz. Poté následuje výše vkladu a pokročilé možnosti určení, o jakou sázku se jedná. Jednou z možností je live sázka, tedy sázka v průběhu zápasu. Další možnosti jako arbitráž či surebet pak využije jen malé procento uživatelů, protože obě možnosti využívají nepoměru kurzů mezi sázkovými kancelářemi a pokud by tyto praktiky používali všichni sázkaři na světě, nebyla by existence sázkových kanceláří vůbec možná, protože by sázkové kanceláře zkrachovaly, nicméně hrstka lidí na této technice profituje. Poslední možností je volba, zda se jedná o sázku s manipulačním poplatkem, či nikoliv. Manipulační poplatek, je nešvar povětšinou českých sázkových kancelářích, který se zrodil v dobách, kdy se sázelo pouze v kamenných pobočkách, ale s nástupem internetových sázkových kanceláří ho některé české sázkové kanceláře stále zavádějí, i přes to, že je snížen na 5% oproti 10%, které jsou v kamenných pobočkách českých sázkových kanceláří. Poté je již možné odeslat tiket.

V menu "Moje sázky" je poté přehled sázek. Sázky si uživatel vyhodnocuje sám. Vzhledem k tomu jak je BetMagnet navržen, není automatické vyhodnocování možné, protože událost a samotný tip je v databázi uložen jako jeden záznam. BetMagnet je tedy vhodný pro sázkaře s širokým spektrem typů sázek. Dále je manuální vyhodnocování výhodné z hlediska rozdílnosti vyhodnocování výsledků v závislosti na pravidlech sázkové kanceláře. Některé sázkové kanceláře například vyhodnocují počet bodů v utkání jen v základním čase a jiné včetně prodloužení. Dalším problémem je získávání dat, které by vedly k automatickému vyhodnocování u exotických lig nebo netypických sázkových příležitostí. Zde nelze automatické vyhodnocování realizovat, protože data k jejich vyhodnocení lze získat jen manuálně. Jedná se například o společenské sázky jako jsou výsledky voleb, či jiných společenských událostí. Dále je zde možnost filtrování sázek dle mnoha kritérií.

Menu "Moje výsledky" je především užitečné pro zjištěné celkové bilance jak si uživatel v sázení vede. Obsahuje také grafy.

BetMagnet je aplikace vhodná pro sázkaře s širokým spektrem sázkových příležitostí, lze přidat libovolný sport či ligu, což je velká výhoda. Obsahuje i grafy s širokými možnostmi třídění. Je k dispozici ve 4 jazycích, ovšem jazyková podpora není dotažená k dokonalosti, protože názvy sportů a lig jsou vždy k uživatelskému účtu přiřazeny jen v jednom jazyce. Uživatelské rozhraní není graficky ztvárněno zrovna nejlépe, což je jedna z velkých nevýhod a mnoho uživatelů může odradit. Další nevýhodou je, že u tiketu neumožňuje přidat poznámky, kde by mohla být například analýza tipu a také neumožňuje přidat důvěru tipu. Jinak lze BetMagnet doporučit jak profesionálním sázkařům, tak hobby sázkařům, i když jeho využití ocení především profesionálnější sázkaři, protože vzhledem k širokým možnostem nastavení může některé hobby sázkaře odradit.

2.2 BetBook.com

BetBook [20] je zaměřen především na komunitu, je velmi, ale opravdu velmi inspirován stránkou FaceBook.com. Nenabízí žádné pokročilé funkce a cílí spíše na sociální sdílení tiketů.

3 Softwarová specifikace požadavků

Softwarová specifikace požadavků (Software requirements specification) popisuje požadavky, které jsou na systém kladeny, konkrétně popisuje všechny akce, které bude uživatel se softwarem provádět. Dělí se na na funkční požadavky (Functional requirements) a nefunkční požadavky (Non-functional requirements).

3.1 Funkční požadavky

Funkční požadavky popisují softwarový systém a jeho komponenty. Definují čeho má softwarový systém dosáhnout pomocí případů užití (Use cases).

Funkční požadavky:

- systém umožní registraci neregistrovanému uživateli
- systém umožní přihlášení/odhlášení registrovanému uživateli
- systém umožní obnovení hesla registrovanému uživateli
- systém umožní změnu hesla přihlášenému uživateli
- systém umožní evidovat sázkové kanceláře přihlášenému uživateli
- systém umožní evidovat ligy přihlášenému uživateli
- systém umožní evidovat tipy přihlášenému uživateli
- systém umožní evidovat tikety přihlášenému uživateli
- systém zobrazí výsledky přihlášeného uživatele
- systém zobrazí veřejné tikety všem uživatelům

3.1.1 Datové funkční požadavky

Specifikují požadavky na data, popisují jaká data chceme uchovávat a jaké výstupy z těchto dat chceme získat.

3.1.1.1 Vstupy Popisují jaká data chceme uchovávat.

U sázkové kanceláře (Bookmaker) budeme zaznamenávat její jednoznačné identifikační číslo (id), jméno (name), měnu (currency), URL (url) a jednoznačné identifikační číslo autora (author_id).

U ligy (League) budeme zaznamenávat její jednoznačné identifikační číslo (id), sport (sport), původ (origin), zkratku (shortcut), jméno (name), URL oficiální webové stránky (official_site_url) a jednoznačné identifikační číslo autora (author_id).

U tipu (Pick) budeme zaznamenávat jeho jednoznačné identifikační číslo (id), datum události (event_date), událost (event), tip (pick), typ typu (pick_type), kurz (odd),

poznámky (notes), status (status), výsledek (result), jednoznačné identifikační číslo ligy (league_id) a jednoznačné identifikační číslo sázkové kanceláře (bookmaker_id).

U tiketu (Ticket) budeme zaznamenávat jeho jednoznačné identifikační číslo (id), vsazenou částku (stake), manipulační poplatek (handling_fee), důvěru v tiket (trust), informaci zda lze tiket zobrazit veřejně (is_public), datum vytvoření (creation_date), počet tipů na tiketu (picks_count), zaplacenou částku (paid), kurz (odd), datum události naposledy rozhodnutého zápasu (event_date_of_last_decided_pick), status (status), zisk (net_profit) a jednoznačné identifikační číslo sázkové kanceláře (bookmaker_id).

U uživatele (User) budeme zaznamenávat jeho jednoznačné identifikační číslo (id), uživatelské jméno (username), křestní jméno (first_name), příjmení (last_name), emailovou adresu (email), heslo (password), zda patří mezi zaměstanance (is_staff), zda je aktivní (is_active), zda je superuživatel (is_superuser), datum posledního přihlášení (last_login), datum registrace (date_joined).

3.1.1.2 Výstupy

Popisují jaká data chceme získat jako výstup ze vstupních dat.

Výpis určitého počtu sázkových kanceláří pro konkrétního uživatele.

Výpis určitého počtu lig pro konkrétního uživatele.

Výpis určitého počtu tipů pro konkrétního uživatele.

Výpis určitého počtu tiketů pro konkrétního uživatele.

Výpis určitého počtu sázkových kanceláří dle konkrétního názvu pro konkrétního uživatele.

Výpis určitého počtu sázkových kanceláří dle konkrétní měny pro konkrétního uživatele.

Výpis určitého počtu lig dle konkrétního sportu pro konkrétního uživatele.

Výpis určitého počtu lig dle konkrétního původu pro konkrétního uživatele.

Výpis určitého počtu tipů dle intervalu zvoleného časového období pro konkrétního uživatele.

Výpis určitého počtu tipů dle konkrétního sportu pro konkrétního uživatele.

Výpis určitého počtu tipů dle konkrétní ligy pro konkrétního uživatele.

Výpis určitého počtu tipů dle konkrétního typu tipu pro konkrétního uživatele.

Výpis určitého počtu tipů dle intervalu zvoleného kurzu pro konkrétního uživatele.

Výpis určitého počtu tipů dle konkrétní sázkové kanceláře pro konkrétního uživatele.

Výpis určitého počtu tipů dle konkrétního statusu pro konkrétního uživatele.

Výpis určitého počtu tiketů dle intervalu zvoleného časového období pro konkrétního uživatele.

Výpis určitého počtu tiketů dle konkrétního typu pro konkrétního uživatele.

Výpis určitého počtu tiketů dle konkrétního důvěry pro konkrétního uživatele.

Výpis určitého počtu tiketů dle intervalu zvoleného kurzu pro konkrétního uživatele.

Výpis určitého počtu tiketů dle konkrétní sázkové kanceláře pro konkrétního uživatele.

Výpis určitého počtu tiketů dle konkrétního statusu pro konkrétního uživatele.

Výpis sumy čistého zisku z uzavřených tiketů, agregovaného dle jednotlivých dní pro konkrétního uživatele.

Událost	Reakce	Aktér
Nová sázková kancelář	Zapiš do seznamu sázk. kanc.	Přihlášený uživatel
Nová liga	Zapiš do seznamu lig	Přihlášený uživatel
Nový tip	Zapiš do seznamu tipů	Přihlášený uživatel
Nový tiket	Zapiš do seznamu tiketů	Přihlášený uživatel
Editace sázkové kanceláře	Edituj sázkovou kancelář	Přihlášený uživatel
Editace ligy	Edituj ligu	Přihlášený uživatel
Editace tipu	Edituj tip	Přihlášený uživatel
Editace tiketu	Edituj tiket	Přihlášený uživatel
Smazání sázkové kanceláře	Smaž sázkovou kancelář	Přihlášený uživatel
Smazání ligy	Smaž ligu	Přihlášený uživatel
Smazání tipu	Smaž tip	Přihlášený uživatel
Smazání tiketu	Smaž tiket	Přihlášený uživatel
Smazání sázkových kanceláří	Smaž sázkové kanceláře	Přihlášený uživatel
Smazání lig	Smaž ligy	Přihlášený uživatel
Smazání tipů	Smaž tipy	Přihlášený uživatel
Smazání tiketů	Smaž tikety	Přihlášený uživatel

Tabulka 1: Tabulka událostí a reakcí

Výpis počtu uzavřených tiketů podle jejich jednotlivých typů pro konkrétního uživatele.

Výpis počtu uzavřených tiketů podle jejich jednotlivých důvěr pro konkrétního uživatele.

Výpis počtu uzavřených tiketů podle jejich jednotlivých statusů pro konkrétního uživatele.

Výpis celkového čistého zisku z uzavřených tiketů pro konkrétního uživatele.

Výpis minimálního, maximálního a průměrného kurzu uzavřených tiketů pro konkrétního uživatele.

Výpis minimální, maximální a průměrné zaplacené částky uzavřených tiketů pro konkrétního uživatele.

Výpis počtu uzavřených tiketů podle jejich statusů.

3.1.1.3 Události a reakce

Specifikace událostí a reakcí jsou v tabulce 1.

3.2 Nefunkční požadavky

Nefunkční požadavky popisují požadavky na design a provedení.

Nefunkční požadavky:

- k implementaci aplikace bude využit programovací jazyk Python s využitím webového frameworku Django, bude se tedy jednat o webovou aplikaci
- aplikace bude multijazyčná

- aplikace bude multiplatformní
- aplikace bude vysoce výkonnostně optimalizovaná

4 Použité technologie

Stručný popis technologií využitých pro tvorbu aplikace.

4.1 Front-end

Technologie vrstvy uživatelského rozhraní.

4.1.1 HTML

HTML (HyperText Markup Language) je značkovací jazyk. Jedná se o hlavní značkovací jazyk webových stránek.

4.1.2 CSS

CSS (Cascading Style Sheets) je jazyk pro popis sémantiky dokumentu napsaného ve značkovacím jazyce.

4.1.3 jQuery

jQuery je JavaScriptová knihovna ulehčující spolupráci mezi HTML a JavaScriptem. Je vydána pod duální licencí MIT (Massachusetts Institute of Technology) a GPL (General Public License).

4.2 Middleware

Technologie střední vrstvy, která je pojítkem mezi vrstvou uživatelského rozhraní a databázovou vrstvou.

4.2.1 Python

Python [5] [1] je objektově orientovaný programovací jazyk jehož autorem je nizozemský programátor Guido van Rossum. Python vznikl v roce 1991 a dnes se těší velké popularitě, je obsažen v distribucích operačního systému Linux.

Python je sice objektově orientovaný, ale umožňuje i procedurální programování, jednou z jeho největších předností oproti jiným jazykům je jednoduchost jeho syntaxe. Python je vyvíjen pod open source licencí.

Aktuálně je Python k dispozici ve verzích 2 a 3.

4.2.2 Django

Django [6] [3] je webový framework, který původně vznikl v novinářském prostředí, ale v roce 2005 byl vydán pod licencí BSD (Berkeley Software Distribution).

Framework Django je napsán v programovacím jazyce Python 2. Momentálně je Django ve verzi 1.4, která nabízí oproti předchozím verzím mnoho pozitivních vylepšení, pro tuto verzi je potřeba minimální verze Pythonu 2.5, 2.6 nebo 2.7.

Django poskytuje mnoho užitečných funkcí, které šetří programátorovi práci tím, že za něj řeší mnohokrát opakované problémy, se kterými se musí potýkat při vývoji webových aplikací.

4.3 Back-end

Technologie databázové vrstvy, zde patří databáze a jiné systémy sloužící k ukládání dat.

4.3.1 PostgreSQL

PostgreSQL [7] je multiplatformní objektově-relační databázový systém. Jeho prvotní historie se datuje od 80. let 20. století a v současnosti je šířen pod licencí MIT (Massachusetts Institute of Technology).

5 Analýza

5.1 Analýza funkčních požadavků

Analýza funkčních požadavků podrobněji popisuje jednotlivé funkční požadavky.

5.1.1 Systém umožní registraci neregistrovanému uživateli

Aby mohl uživatel využívat funkčnost, kterou aplikace poskytuje, bude se muset nejprve zaregistrovat. Uživatel si zvolí uživatelské jméno a heslo a poskytne také svou emailovou adresu, na kterou budou v případě ztráty hesla odeslány pokyny k jeho obnovení. Registrační formulář bude ošetřen proti automatizovaným robotům. Proces registrace bude obsahovat také proces ověření emailové adresy. Systém registrovanému uživateli vytvoří uživatelský účet. Uživatelské jméno a email bude v celém systému unikátní.

5.1.2 Systém umožní přihlášení/odhlášení registrovanému uživateli

Registrovaný uživatel se bude muset přihlásit do systému pomocí uživatelského jména a hesla, aby mohl využívat všech funkcí, které systém poskytuje. Součástí registračního formuláře bude i odkaz na stránku s informacemi jak postupovat v případě ztráty hesla.

5.1.3 Systém umožní obnovení hesla registrovanému uživateli

V případě, že dojde ke ztrátě hesla, uživatel požádá o zaslání pokynů k obnově hesla na emailovou adresu, kterou zadal při registraci. Email bude obsahovat odkaz na obnovu hesla, po navštívení odkazu uživatel zadá (dvakrát z důvodu kontroly) nové heslo. A poté bude moci uživatel využívat nové heslo.

5.1.4 Systém umožní změnu hesla přihlášenému uživateli

Přihlášený uživatel si bude moci změnit heslo, ať už k tomu bude mít jakékoliv důvody.

5.1.5 Systém umožní evidovat sázkové kanceláře přihlášenému uživateli

Přihlášený uživatel bude moci vložit, editovat a smazat sázkovou kancelář. Dále bude moci třídit vložené sázkové kanceláře dle zvolených kritérií.

5.1.6 Systém umožní evidovat ligy přihlášenému uživateli

Přihlášený uživatel bude moci vložit, editovat a smazat ligu. Dále bude moci třídit vložené ligy dle zvolených kritérií.

5.1.7 Systém umožní evidovat tipy přihlášenému uživateli

Přihlášený uživatel bude moci vložit, editovat a smazat tip. Dále bude moci třídit vložené tipy dle zvolených kritérií.

5.1.8 Systém umožní evidovat tikety přihlášenému uživateli

Přihlášený uživatel bude moci vložit, editovat a smazat tiket. Dále bude moci třídit vložené tikety dle zvolených kritérií.

5.1.9 Systém zobrazí výsledky přihlášeného uživatele

Systém zobrazí přihlášenému uživateli výsledky jeho dosavadního tipování jak v textové tak v grafické podobě pomocí různých druhů grafů.

5.1.10 Systém zobrazí veřejné tikety všem uživatelům

Tikety, které zvolí uživatel jako veřejné obdrží URL pomocí níž bude možné tiket sdílet.

5.2 Diagramy případů užití

Diagramy případů užití zobrazují chování systému z hlediska uživatele, diagramy se nacházejí v příloze A.

K vytvoření diagramů byl využit modelovací jazyk UML (Unified Modeling Language) [2].

Obrázek 6 zobrazuje aktéry.

Obrázek 7 zobrazuje případy užití:

- systém umožní registraci neregistrovanému uživateli
- systém umožní přihlášení/odhlášení registrovanému uživateli
- systém umožní obnovení hesla registrovanému uživateli
- systém umožní změnu hesla přihlášenému uživateli

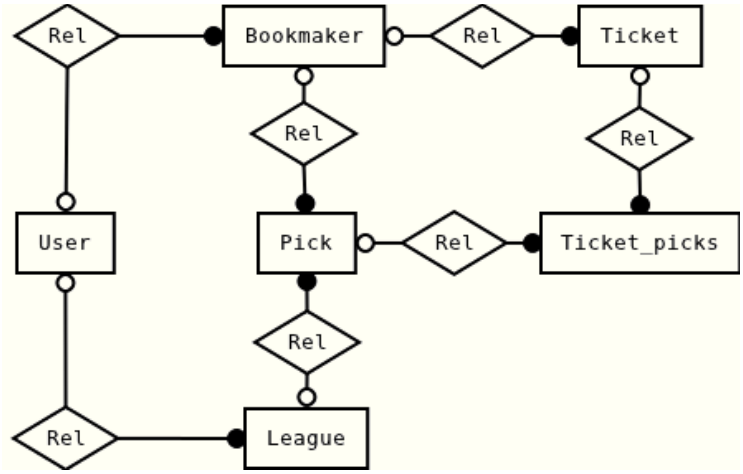
Obrázek 8 zobrazuje případ užití: systém umožní evidovat sázkové kanceláře přihlášenému uživateli.

Obrázek 9 zobrazuje případ užití: systém umožní evidovat ligy přihlášenému uživateli.

Obrázek 10 zobrazuje případ užití: systém umožní evidovat tipy přihlášenému uživateli.

Obrázek 11 zobrazuje případy užití:

- systém umožní evidovat tikety přihlášenému uživateli
- systém zobrazí výsledky přihlášeného uživatele
- systém zobrazí veřejné tikety všem uživatelům



Obrázek 1: ERM s vyznačením vazby

5.3 Datová analýza

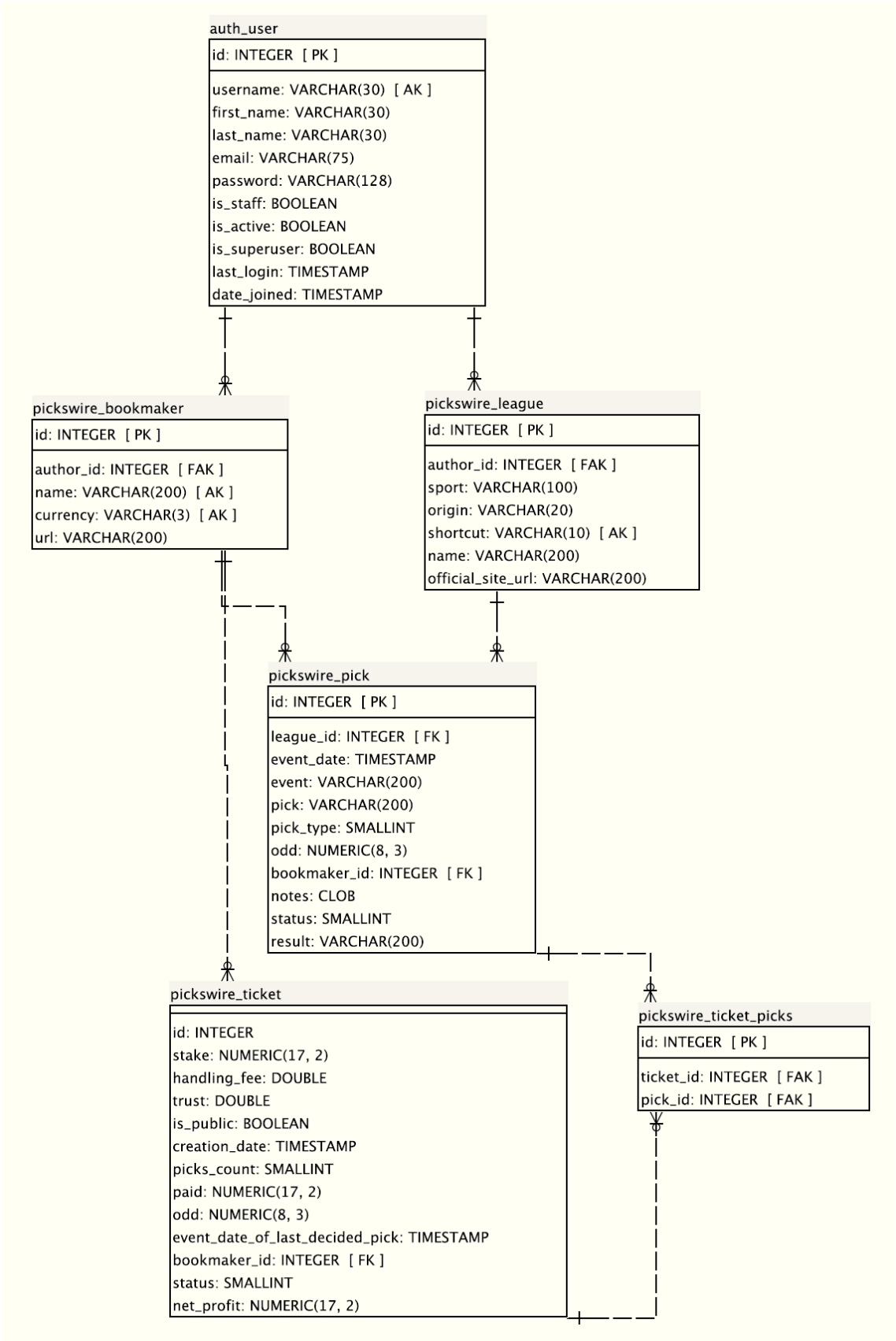
5.3.1 Lineární zápis seznamů typů entit a jejich atributů

Poznámka 5.1 primární klíč je označen podtržením a cizí klíče jsou označeny tučnou kurzívou

Bookmaker(id, name, currency, url, ***author_id***)
League(id, sport, origin, shortcut, name, official_site_url, ***author_id***)
Pick(id, event_date, event, pick, pick_type, odd, notes, status, result, ***bookmaker_id***, ***league_id***)
Ticket(id, stake, handling_fee, trust, is_public, creation_date, picks_count, paid, odd, event_date_of_last_decided_pick, status, net_profit, ***bookmaker_id***)
Ticket_picks(id, ***ticket_id***, ***pick_id***)
User(id, username, first_name, last_name, email, password, is_staff, is_active, is_superuser, last_login, date_joined)
UserRelBookmaker(User, Bookmaker) 1:N
UserRelLeague(User, League) 1:N
BookmakerRelPick(Bookmaker, Pick) 1:N
BookmakerRelTicket(Bookmaker, Ticket) 1:N
LeagueRelPick(League, Pick) 1:N
PickRelTicket_picks(Pick, Ticket_picks) 1:N
TicketRelTicket_picks(Ticket, Ticket_picks) 1:N

5.3.2 ER diagramy

Obrázek 1 zobrazuje ER diagram s vyznačením vazby mezi entitami a obrázek 2 zobrazuje entity i s jejich atributy (včetně datových typů).



Obrázek 2: ERM s atributy

5.3.3 Datový slovník

Úplné tabulky atributů se nacházejí v příloze B.

Tabulka 2 (Bookmaker) eviduje sázkovou kancelář. Měna (currency) obsahuje na výběr celkem 18 měn, do databáze bude uložen kód ve formátu ISO 4217. URL (url) může zůstat nevyplněno například z důvodu, že si někdo v kamenné pobočce sázkové kanceláře, která nemá webové stránky a přes to bude chtít tyto sázky evidovat. Atributy author_id, name a currency jsou dohromady unikátní.

Tabulka 3 (League) eviduje ligu. Sport (sport) obsahuje na výběr mnoho sportů. Původ (origin) obsahuje na výběr 247 států (některé z nich nejsou státy, například Evropská Unie), většina z nich splňuje ISO 3166-1 alpha-2. Atributy author_id a shortcut jsou dohromady unikátní.

Tabulka 4 (Pick) eviduje tip. Typ tipu (pick_type) obsahuje na výběr předem vybrané typy sázek. Status (status) obsahuje možnosti živý, výhra, prohra a storno.

Tabulka 5 (Ticket) eviduje tiket. Manipulační poplatek (handling_fee) obsahuje výběr možnosti Ne, 5% a 10 %. Důvěra (trust) obsahuje na výběr možnosti 1/5, 2/5, 3/5, 4/5 a 5/5. Název atributu event_date_of_last_decided_pick byl v tabulce zkrácen na edoldp.

Tabulka 6 (Ticket_picks) eviduje je vazební tabulka pro M:N relaci mezi tikety a tipy. Atributy ticket_id a pick_id jsou dohromady unikátní.

Tabulka 7 (User) eviduje uživatele. Atribut username je unikátní.

6 Implementace

Tato kapitola popisuje implementaci důležitých částí aplikace.

Zdrojový kód aplikace je kompletně v angličtině včetně komentářů a označení překladových řetězců.

Zdrojové kódy jsou jako příloha jsou na DVD jehož obsah je v příloze C.

6.1 Databázové modely

Framework Django využívá vlastní objektově relační mapování (ORM). Objektově relační mapování zajistí konverzi dat mezi databází a programovacím jazykem (objektově orientovaným).

Návrh databáze je pro celou aplikaci to nejdůležitější.

Databázové modely se dle konvencí frameworku Django nacházejí v souboru `models.py`.

Pokud programátor nedefinuje primární klíč, framework Django jej automaticky vytvoří u každého modelu a nazve jej `id`. Takto vytvořený primární klíč dostává automaticky také databázový index.

Databázový index také automaticky dostávají cizí klíče.

Pro vytváření databázových dotazů pomocí ORM frameworku Django je potřebná kompletní znalost jeho funkčnosti, v opačném případě může dojít k vytváření značně náročných a neoptimalizovaných dotazů.

Tato aplikace je založena především na práci s daty, databáze je tedy to nejdůležitější. Do databáze se bude velkou měrou jak zapisovat, tak číst. Čtení ovšem bude představovat větší podíl práce s daty. Z tohoto důvodu je důležitý vhodný návrh databázových indexů. Databázové indexy je třeba používat s rozvahou, protože počet řádků bude v databázi rychle narůstat a přílišný počet indexovaných polí by mohl způsobit výrazné zpomalení databáze při vkládání dat.

6.1.1 Model Bookmaker

Model `Bookmaker` představuje účet u sázkové kanceláře.

Každý uživatel si může vytvořit libovolný počet účtů v různých měnách. Může tak například mít účet u stejné sázkové kanceláře, ale každý v jiné měně.

```
class Bookmaker(models.Model):

    CURRENCY_CHOICES = (
                    (u'USD', _(u'U.S. Dollar')),
    )

    author = models.ForeignKey(User, editable=False)
    name = models.CharField(_(u'name'), max_length=200, db_index=True)
    currency = models.CharField(_(u'currency'), choices=CURRENCY_CHOICES, max_length=3,
        db_index=True)
    url = models.URLField(u'URL', blank=True, help_text=_(u'For safety reasons use https:// if
        availabile.'))
```

Výpis 1: Ukázka části třídy modelu Bookmaker

Poznámka 6.1 V ukázce zdrojového kódu jsou vynechány komentáře a obsah CURRENCY_CHOICES byl zkrácen z důvodu přílišné délky.

Model `Bookmaker` obsahuje následující pole: `author`, `name`, `curency` a `URL`.

Pole `autor` je cizí klíč modelu `User`. Tento model je již částí frameworku Django, není tedy potřeba je pro potřeby aplikace `pickswire` vytvářet. Atribut `editable=False` zajišťuje, že se toto pole neobjeví ve formulářích pro přidávání a editaci modelu, místo toho se toto přiřazení vyřeší v administrační třídě modelu přetížením metody `save()`. Django automaticky přidá databázový index na cizí klíče, není tedy nutné dále definovat přidání cizího klíče.

Pole `name` představuje název sázkové kanceláře a bude mít databázový index, protože pomocí něj se bude často třídit a vyhledávat.

Pole `currency` bude uchovávat informace o zvolené měně. Měna bude uživateli nabídnuta jako rozevírací seznam. Do tohoto seznamu bylo vybráno několik důležitých měn. V databázi budou uloženy pouze jejich zkratky, které splňují normu ISO 4217. V přehledu pak budou u měn zobrazeny i ikony vlajek. K tomu byly využity ikony ze stránky FAMFAMFAM.com, jejichž licence to umožňuje. Měna bude mít také databázový index, protože se pomocí ní bude také často třídit.

Pole `URL` nemusí uživatel zadávat, například z důvodu, že si vsadí v síti sázkových kanceláří, které mají pouze kamenné pobočky. Pokud však uživatel přesto URL zadá, doporučuje se zadávat s šifrovaným protokolem. Tento pomocný text se zobrazí ve formulářích pro přidávání a editaci.

Pole `autor`, `name` a `currency` budou dohromady unikátní.

6.1.2 Model League

Model `League` bude představovat sportovní ligu či turnaj.

```
class League(models.Model):

    SPORT_CHOICES = (
                (u'soccer', _(u'Soccer')),

    )

    ORIGIN_CHOICES = (
                    (u'ad', u'Andorra'),
    )

    author = models.ForeignKey(User, editable=False)
    sport = models.CharField(_(u'sport'), choices=SPORT_CHOICES, max_length=100, db_index=
        True)
    origin = models.CharField(_(u'origin'), choices=ORIGIN_CHOICES, max_length=20)
    shortcut = models.CharField(_(u'shortcut'), max_length=10, db_index=True)
    name = models.CharField(_(u'name'), max_length=200, db_index=True)
     official_site_url = models.URLField(_(u'official site'), blank=True)
```

Výpis 2: Ukázka části třídy modelu League

Poznámka 6.2 V ukázce zdrojového kódu jsou vynechány komentáře a obsah SPORT_CHOICES a ORIGIN_CHOICES byl zkrácen z důvodu přílišné délky.

Model `League` obsahuje následující pole: `author`, `sport`, `origin`, `shortcut`, `name` a `official_site_url`.

Pole `autor` je stejně jako v případě modelu `Bookmaker` cizí klíč modelu `User`. Jak již bylo vysvětleno u předchozího modelu, databázový index bude vytvořen automaticky. Pro přiřazení platí opět to samé jako u předchozího modelu, tedy přetížení metody `save()`. Každý uživatel si vytvoří libovolný počet lig.

Pole `sport` představuje sport. Jedná se o nabídku sportů, které jsou již součástí aplikace, není nutné, aby uživatel vytvářel vlastní sporty. V ukázce zdrojového kódu je pro přílišnou délku zobrazen jen jeden sport.

Pole `sport` představuje původ ligy v nabídce přes 250 států, včetně kontinentů jako například Evropa. V ukázce zdrojového kódu je pro přílišnou délku zobrazen jen jeden původ.

Pole `shortcut` představuje zkratku ligy, zkratka ligy je potřebná především kvůli zobrazení na webu ve chvílích, kdy se název ligy nehodí z důvodu jeho přílišné délky.

Pole `name` přestavuje jméno ligy.

Pole `official_site_url` přestavuje oficiální URL ligy.

Pole `autor` a `shortcut` jsou dohromady unikátní.

6.1.3 Model Pick

Model `Pick` již pro aplikaci představuje větší důležitost než předchozí dva popisované modely. Tento model totiž přestavuje tip a uživatel aplikace bude především vkládat tipy a sestavovat tikety. U předchozích modelů se totiž předpokládá, že uživatel s nimi bude pracovat jen při jejich vytvoření, které oproti tomuto modelu budou méně časté.

```
class Pick(models.Model):

    PICK_TYPE_CHOICES = (
        (1, _(u'Money Line')),

    )

    PENDING_STATUS = 1
    WON_STATUS = 2
    LOSS_STATUS = 3
    PUSH_STATUS = 4
    STATUS_CHOICES = (
        (PENDING_STATUS, _(u'Pending')),
        (WON_STATUS, _(u'Won')),
        (LOSS_STATUS, _(u'Loss')),
        (PUSH_STATUS, _(u'Push')),
    )

    league = models.ForeignKey(League)
    event_date = models.DateTimeField(_(u'event date'), db_index=True)
    event = models.CharField(_(u'event'), max_length=200)
    pick = models.CharField(_(u'pick'), max_length=200)
    pick_type = models.PositiveSmallIntegerField(_(u'pick type'), choices=PICK_TYPE_CHOICES)
    odd = models.DecimalField(_(u'odd'), max_digits=8, decimal_places=3, validators=[
        MinValueValidator(1.001)], help_text=_(u'Decimal format'))
    bookmaker = models.ForeignKey(Bookmaker)
    notes = models.TextField(_(u'notes'), blank=True)
    status = models.PositiveSmallIntegerField(_(u'status'), choices=STATUS_CHOICES, default
        =1, editable=True, db_index=True)
    result = models.CharField(_(u'result'), max_length=200, blank=True)
```

Výpis 3: Ukázka části třídy modelu Pick

Poznámka 6.3 V ukázce zdrojového kódu jsou vynechány komentáře a obsah PICK_TYPE_CHOICES byl zkrácen z důvodu přílišné délky.

Model `Pick` obsahuje následující pole: `league`, `event_date`, `event`, `pick`, `pick_type`, `odd`, `bookmaker`, `notes`, `status`, `status` a `result`.

Pole `league` je cizí klíč modelu `League` a databázový index bude opět vytvořen automaticky.

Pole `event_date` představuje datum a čas události. Vzhledem k tomu, že data budou primárně tříděna dle tohoto kritéria, bude obsahovat databázový index.

Pole `event` představuje název události.

Pole `pick` představuje tip.

Pole `pick_type` pak představuje typ tipu, což je ovšem rozevírací seznam, ze kterého si uživatel vybere, `pick` a `pick_type` se liší tím, že pokud uživatel zadá jako tip například: Počet bodů - méně než 185, tak typ tipu bude Počet bodů - méně, z rozdílů je tedy patrný rozdíl tedy, že typ tipu již neobsahuje hranici počtu bodů.

Pole `odd` představuje kurz ve formátu, který je užíván především v Evropě a přesnost je tři desetinná místa, přičemž minimální hodnota je 1,001.

Pole `bookmaker` je cizí klíč modelu `Bookmaker` a uživatel vždy vybere, u které sázkové kanceláře je tento tip s tímto daným kurzem.

Pole `notes` přestavuje poznámky, které mohou obsahovat buď poznámky nebo analýzu daného tipu, tuto funkčnost ocení především zkušení sázkaři.

Pole `status` přestavuje status tipu a může nabývat čtyř hodnot. Ihned po vytvoření tipu má tip status živý a poté jej může uživatel vyhodnotit jako výhra, prohra nebo storno. Tato funkčnost je důležitá z hlediska automatického rozhodování tiketů na základě rozhodnutých tipů, ovšem bude popsána později.

Pole `result` přestavuje výsledek dané události tipu, ale není povinné, protože některým sázkařům stačí pouze informace o statusu tipu.

6.1.4 Model Ticket

Model `Ticket` je úzce spjat s modelem `Pick`, mezi nimi je vazba M:N tedy, že jeden tip může být na více tiketech a stejně tak jeden tiket může obsahovat více tipů.

```
class Ticket(models.Model):

    TRUST_CHOICES = (
        (0.2, u'1/5'),

    )

    HANDLING_FEE_CHOICES = (
        (0.00, _(u'No')),
        (0.05, u'5%'),
        (0.10, u'10%'),
    )

    PENDING_STATUS = 1
    WON_STATUS = 2
    LOSS_STATUS = 3
    PUSH_STATUS = 4
    STATUS_CHOICES = (
                    (PENDING_STATUS, _(u'Pending')),
                    (WON_STATUS, _(u'Won')),
                    (LOSS_STATUS, _(u'Loss')),
                    (PUSH_STATUS, _(u'Push')),
    )

    picks = models.ManyToManyField(Pick, verbose_name=_(u'picks'), help_text=_(u'You can add
        all picks with "pending" status with same bookmaker only.'))
    stake = models.DecimalField(_(u'stake'), max_digits=17, decimal_places=2)
    handling_fee = models.FloatField(_(u'handling fee'), choices=HANDLING_FEE_CHOICES,
        default=0.00, help_text=_(u'Handling fee is commonly used by czech bookmakers.'))
    trust = models.FloatField(_(u'trust'), choices=TRUST_CHOICES)
    is_public = models.BooleanField(_(u'is public'), help_text=_(u'Public tickets get public URL
        so you can share them with friends.'))

    creation_date = models.DateTimeField(_(u'creation date'), auto_now_add=True, editable=False)

    picks_count = models.PositiveSmallIntegerField(_(u'picks count'), default=0, editable=False)
    paid = models.DecimalField(_(u'paid'), max_digits=17, decimal_places=2, editable=False)
    odd = models.DecimalField(_(u'odd'), max_digits=8, decimal_places=3, validators=[
        MinValueValidator(1.001)], editable=False)
    event_date_of_last_decided_pick = models.DateTimeField(_(u'event date of last decided pick'),
        editable=False, db_index=True)
    bookmaker = models.ForeignKey(Bookmaker, editable=False)

    status = models.PositiveSmallIntegerField(_(u'status'), choices=STATUS_CHOICES, default
        =1, editable=False, db_index=True)
    net_profit = models.DecimalField(_(u'net profit'), max_digits=17, decimal_places=2, editable=
        False, default=0)
```

Výpis 4: Ukázka části třídy modelu Ticket

Poznámka 6.4 V ukázce zdrojového kódu jsou vynechány komentáře a obsah TRUST_CHOICES byl zkrácen z důvodu přílišné délky.

Model `Ticket` obsahuje následující pole: `picks`, `stake`, `handling_fee`, `trust`, `is_public`, `creation_date`, `picks_count`, `paid`, `odd`, `event_date_of_last_decided_pick`, `bookmaker`, `status`, `net_profit`.

Pole `picks` obsahuje tipy. Jak již bylo popsáno výše, jedná se o spojení M:N.

Pole `stake` představuje sázku, tedy výši finančních prostředku vsazených na tiket, ale bez započtení manipulačního poplatku.

Pole `handling_fee` představuje manipulační poplatek, který je především nešvarem českých sázkových kanceláří a na internetu bývá obvykle 5% ze vsazené částky a v kamenných pobočkách obvykle 10%. Některé české sázkové kanceláře mají i vyjímky, že od určité výše kurzu tiketu se manipulační poplatky neplatí vůbec. Uživatel aplikace má na výběr, buď žádný manipulační poplatek, 5% nebo 10%, ale v databázi je tato hodnota uložena v procentech.

Pole `trust` představuje důvěru sázkaře v tiket. Uživatel aplikace má na výběr z pěti možností, ale v databázi je toto pole uloženo v procentech.

Pole `is_public` označuje, zda se jedná o veřejný tiket nebo ne. Veřejný tiket je k vidění i pro neregistrované uživatele. Tato funkčnost tedy slouží uživatelům, kteří chtějí sdílet obsah svých tiketů s ostatními. Přístup k veřejnému tiketu lze provést přes URL veřejného tiketu, ta se zobrazí v případě, že je tiket veřejný.

Pole `creation_date` označuje datum a čas vytvoření tiketu.
Atribut `auto_now_add=True` zajistí, že framework Django automaticky přiřadí datum vytvoření instance objektu `Ticket` při jejím vložení do databáze.

Obsah následujících pěti polí nezadává uživatel aplikace, ale aplikace jejich obsah nastaví sama na základě dat vložených uživatelem.

Pole `picks_count` představuje počet tipů vložených na tiket. Počet tipů vložených na tiket je potřeba uložit do databáze, i přes to, že to lze zjistit databázovým dotazem, ale vzhledem k tomu, že v administračním rozhraní bude mít uživatel najednou zobrazeny desítky, ale i jednotky stovek záznamů, musel by být proveden dotaz pro každý jeden tiket. Z tohoto důvodu je lepší uložit tuto hodnotu do databáze.

Zde ovšem nastává problém s ORM frameworku Django. U relací M:N mezi objekty totiž framework Django ukládá záznamy o relaci M:N až po provedení signálu `post_save`. Signál ve frameworku Django se odesílá na základě změny na instanci nějakého objektu. Po uložení instance objektu `Ticket` nelze v tomto signálu tedy `post_save` provést dotaz, který by spočítal počet relací M:N, protože v databázi je uložena pouze instance modelu `Ticket`. Proto jsem tento problém vyřešil spočtením počtu tipů z dat z formuláře vkládání.

Pole `paid` představuje celkovou výši finančních prostředků vsazených na tiket, tedy včetně manipulačního poplatku.

Pole `odd` představuje celkový kurz tiketu. Tedy v případě, že je na tiketu více tipů, tak se kurzy všech tipů vynásobí.

Pole `event_date_of_last_decided_pick` přestavuje datum a čas naposledy rozhodnuté události na tiketu. Tedy pokud je na tiketu více tipů, tak zde bude uloženo datum

události tipu, který bude rozhodnut naposledy. Tikety budou dle tohoto data primárně seřazeny, proto bude mít toto pole databázový index.

Pole `bookmaker` je cizí klíč modelu `Bookmaker` a bude přiřazen automaticky. Je potřeba zajistit, aby všechny tipy vložené na tiket měly stejný účet u sázkové kanceláře, tedy stejnou instanci modelu `Bookmaker`, proto je potřeba ošetřit formulář pro vkládání tiketu validací, která zkontroluje, zda se všechny tipy (v případě, že je jich více) mají stejnou instanci modelu `Bookmaker`, pokud ne, bude vyvolána validační chyba formuláře a formulář nepůjde uložit, dokud uživatel chybu neopraví. Dle konvencí frameworku Django se formuláře nacházejí v souboru `forms.py`.

Poslední dvě pole, které model obsahuje, budou vyhodnocovány automaticky v závislosti na změně instancí modelu `Pick`. Jedná se o pole `status`, které může stejně jako u modelu `Pick` nabývat čtyř hodnot, tedy ihned po vytvoření tiketu má tiket status živý a poté může být automaticky vyhodnocen jako výhra, prohra nebo storno. A jako poslední je pole `net_profit`, které představuje čistý zisk.

Vyhodnocování tiketů tak může proběhnout automaticky na základě rozhodnutých tipů, které tiket obsahuje. Framework Django k tomu využívá takzvané signály, ty budou popsány v samostatné kapitole.

6.1.5 Optimalizace databázových dotazů

Jak již bylo zmíněno výše, tak v případě neznalosti fungovaní ORM, mohou vzniknout neoptimalizované dotazy.

Všechny dotazy této aplikace byly optimalizovány včetně optimalizace dotazů, které vytváří framework Django, protože aplikace využívá jeho administračního rozhraní.

Je potřeba poznamenat, že dotazování je tzv. líné, tedy že dotaz se provede až tehdy, je-li to potřeba.

V případě, že bylo potřeba spojit více tabulek, bylo využito funkce `select_related()`, která spojí tabulky pomocí vnitřního spojení, provede se tak pouze jeden dotaz.

Dále funkce `defer()` a `only()` zabraňují načítání polí, které není potřeba zobrazovat.

Užitím funkce `select_for_update()` v souboru `signals.py` se všechny instance objektu `Ticket` zamknou pro aktualizaci, dokud transakce neskončí. Na databázové vrstvě to tedy znamená, že se zamknou řádky.

6.2 Signály

Signály frameworku Django umožňují vyslání upozornění, že někde v aplikaci nastala nějaká událost.

V případě aplikace `pickswire` budou signály využity vždy po uložení instance modelu `Pick`. Metoda vybere všechny tikety, na kterých se daný tip nachází, a zkontroluje všechny statusy tipů, které se na tiketu nachází a na základě statusu těchto tipů se tak nastaví status celého tiketu. Zároveň také vypočte čistý zisk.

```
def pick_post_save_handler(sender, instance, **kwargs):
    tickets = Ticket.objects.select_for_update().order_by().filter(picks__id = instance.id)
    for ticket in tickets:
        picks_from_ticket = Pick.objects.defer('league', 'event', 'pick', 'pick_type', 'odd', '
            bookmaker', 'notes', 'result').filter(ticket__id = ticket.id).order_by('-event_date')
        lst_statuses = []
        for pick_from_ticket in picks_from_ticket:
            lst_statuses.append(pick_from_ticket.status)
        if Ticket.PENDING_STATUS in lst_statuses:
            ticket.status = Ticket.PENDING_STATUS
            ticket.net_profit = 0
        elif Ticket.LOSS_STATUS in lst_statuses:
            ticket.status = Ticket.LOSS_STATUS
            ticket.net_profit = -ticket.paid
        elif (Ticket.PENDING_STATUS not in lst_statuses) and (Ticket.LOSS_STATUS not in
            lst_statuses):
            if (Ticket.WON_STATUS in lst_statuses):
                ticket.status = Ticket.WON_STATUS
                ticket.net_profit = (ticket.get_decided_ticket_odd() * ticket.stake) - ticket.paid
            else:
                ticket.status = Ticket.PUSH_STATUS
                ticket.net_profit = 0
        ticket.save()
post_save.connect(pick_post_save_handler, sender=Pick)
```

Výpis 5: Ukázka části soubou signals.py

Další signál je využit v případě, že dojde ke smazání tipu. Protože v případě, že jsou modely spojeny pomocí cizího klíče framework Django v administračním rozhraní při smazání instance modelu zobrazí, že budou smazány i instance jiného modelu, které mají na tuto instanci vazbu pomocí cizího klíče. Takže v případě vazby 1:N dojde ke kaskádovitému smazání.

Ovšem v případě vazby M:N se tak neděje. Z tohoto důvodu je použit signál, jehož metoda po smazání instance modelu `Pick` smaže i instance modelu `Ticket`, které mají vazbu instanci modelu `Pick`.

6.3 Administrační rozhraní

Součástí frameworku Django je také automaticky vytvořené administrační rozhraní vygenerované na základě vytvořených databázových modelů. Toto rozhraní je však značně

univerzální, a proto je potřeba jej upravit a vytvořit nad něj jakousi nadstavbu pro potřeby aplikace `pickswire`.

U všech modelů aplikace `pickswire` je potřeba přetížit metodu `queryset()`, aby každý uživatel měl zobrazeny pouze instance modelů, které sám vytvořil. To samé je potřeba provést pro formulářová pole u modelu `Pick`, pro pole `league` a `bookmaker` a u modelu `Ticket` pro pole `picks`.

U modelů `Bookmaker`, `League` a `Ticket` je přetížena metoda `save_model` z důvodů, které jsou popsány v kapitolách o vytváření databázových modelů, tedy z důvodů, kdy je potřeba doplnit některá pole na základě dat vložených uživatelem, ale také automatické vytvoření vazby mezi instancí uživatele a instancí objektu, jehož metoda je přetěžována.

Administrační rozhraní frameworku Django primárně slouží především pro přístup spíše administrátorům, ale v případě této aplikace jej budou využívat všichni uživatelé, kteří pomocí něj budou data nejen vkládat, ale také zobrazovat, proto je potřeba upravit jeho primární chování. Například změnit odkazy v levém sloupci tak, aby se zobrazily informace o instanci nikoliv editace instance jako je tomu v případě základního nastavení a samozřejmě vytvořit nový odkaz na editaci instance.

6.3.1 Vlastní filtry

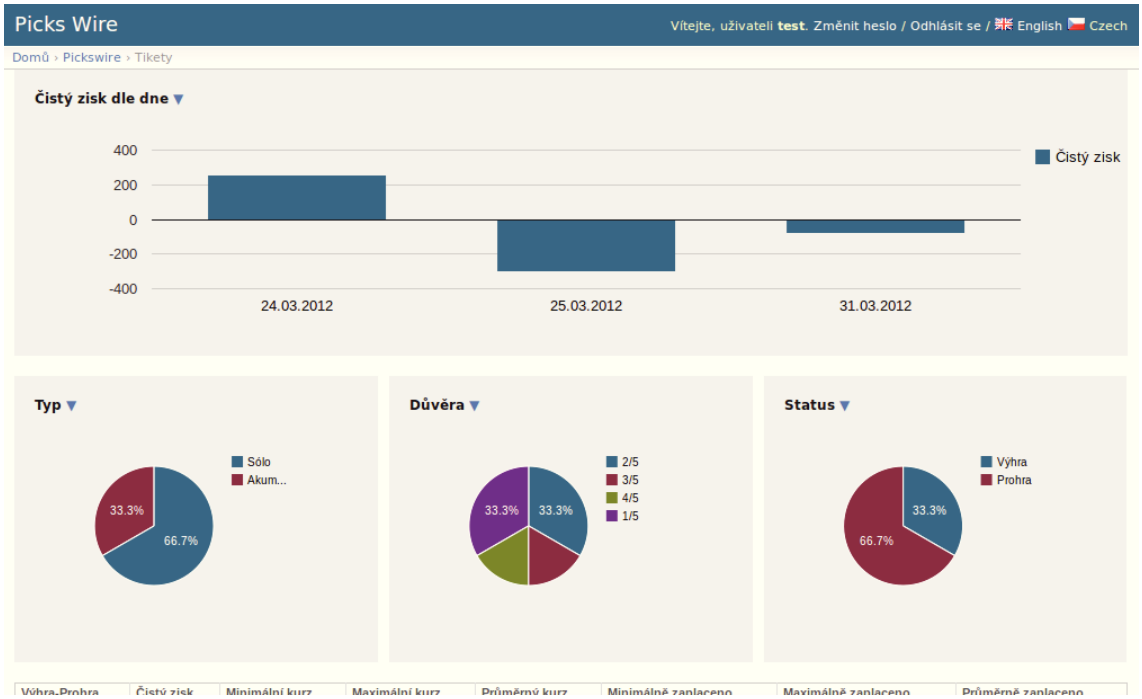
Další přizpůsobení administrace spočívá ve vytvoření vlastních filtrů. Framework Django sice umožňuje snadné vytvoření filtrů v administračním rozhraní ovšem pro potřeby této aplikace je potřeba i další filtry a ani základní filtry nemohou správně fungovat. Je potřeba filtrovat opět pouze na základě instancí, které vytvoří uživatel a dále pro lepší přehlednost jsem filtry upravil tak, že pokud neexistuje instance, která by obsahovala vazbu na nějaké další instance, tak nebude ve filtru zobrazena kvůli větší přehlednosti, protože logicky pokud instance ve filtru nemá vazby na žádné další instance, tak pokud by se tento filtr aplikoval stejně by vrátil nulový počet záznamů.

Framework Django od verze 1.4 umožňuje vytvořeních vlastních filtrů, dle konvencí se tyto filtry nachází v souboru `list_filters.py`. Vytvořeno bylo mnoho filtrů, protože filtrování dat na základě různých kritérií je jednou z nejdůležitějších funkcí této aplikace. U podobných filtrů bylo využito dědičnosti.

Další úprava jsou takzvané `actions` pro administraci modelu `Pick`. Uživatel má díky nim možnost rozhodnout najednou více tipů zároveň, jednoduše zaškrtne všechny tipy, které chce zahrnout do výběru pro společnou akci a zvolí akci. Akce jsou celkem tři a umožňují rozhodnout tipy jako vyhrané, prohrané nebo stornované. Akce tedy mění pole `status` tipu.

Bohužel vzhledem k tomu, že framework Django nespustí signál v případě uložení více instancí objektu zároveň, je potřeba projít všechny záznamy cyklem a uložit každý zvlášť, což je pomalejší.

Dále proběhly různé druhy drobných úprav jako například zabránění editace již rozhodnutých tiketů, nebo různé metody, které formátují text do požadovaného formátu pro následný výpis v administračním rozhraní.



Obrázek 3: Grafické přehledy

6.3.2 Grafické přehledy

Grafické přehledy se nacházejí v admnistraci modelu `Ticket`.

Pro vykreslení grafů byly využity Google Chart Tools [8]. Šablona byla vyděděna z původní administrační šablony frameworku Django, protože se je to opět konvenční doporučovaný postup.

Pro lepší přehlednost je možné grafy pomocí tlačítka schovat a poté zpět zobrazit. Při úvodním načtení stránky jsou grafy chované a uživatel si je může buď zobrazit nebo nechat schované. Vzhledem k tomu, že komunikace s HTTP probíhá takovým způsobem, že i při opakovaném požadavku od klienta server s klientem komunikuje jako by to bylo poprvé a zobrazování grafů je realizováno pomocí Java Scriptového frameworku jQuery, je potřeba zařídit, aby si server pamatoval, zda uživatel chce mít grafy zobrazeny či ne, je potřeba využít cookies. Cookie pošle HTTP server klientovi, který jej uloží v počítači uživatele. Vzhledem k tomu, že framework jQuery neumožňuje práci s cookies, využil jsem rozšíření pro jQuery s názvem jQuery cookie a pro práci se zobrazováním a schováváním elementu grafu rozšíření jQuery showhide [9], které spolupracuje právě s jQuery cookie [10], obě rozšíření pro svou práci samozřejmě potřebují framework jQuery, který je již obsažen v administrační aplikaci frameworku Django.

Grafy jsou celkem čtyři. Graf zisků zobrazuje čistý zisk za jednotlivé dny, jedná se o sloupcový graf. Další tři grafy jsou koláčové a zobrazují počet tiketů dle kritérií typ, důvěra a status. Grafy lze vidět na obrázku 3.

6.3.3 Dodatečné pohledy a šablony

Administrační rozhraní bylo rozšířeno o pohledy a k nim tedy i patřičné šablony, které jsou dostupné pouze uživatelům přihlášeným do administračního rozhraní.

Jedná se o zobrazení pro jednotlivou instanci modelu `Pick` a `Ticket`.

Samozřejmě je nutné také přidat nastavení URL, které se dle konvencí frameworku Django nachází v souboru `urls.py` v jednotlivých aplikacích, či projektech.

6.4 Autentizační a Autorizační systém

Jednou z mnoha výhod frameworku Django je autentizační a autorizační systém. Tento systém obsahuje:

- Uživatele
- Práva: binární (ano/ne), říká, zda má uživatel oprávnění k provedení dané úlohy
- Skupiny: umožňuje hromadné přiřazení uživatelských práv
- Zprávy: umožňuje posílání zpráv mezi uživateli

O autentizační proces se ve frameworku Django stará aplikace `django.contrib.auth`.

V aplikaci pro evidenci sázkařských tipů je autentizace uživatele nutností, aby bylo možné svázat jeho uživatelský účet s funkcemi, které nabízí aplikace pro evidenci sázkařských tipů.

Ovšem nejprve je nutné uživatelský účet vytvořit a přiřadit mu oprávnění k přihlášení a správě aplikace pro evidenci sázkařských tipů.

Django nabízí například jednoduché pohledy `django.contrib.auth.views.login` a `django.contrib.auth.views.logout`, ke kterým už stačí jen vytvořit šablony a uživatel se může přihlásit a odhlásit. Ovšem aplikace pro evidenci sázkařských tipů obsahuje operace CRUD v administračním systému frameworku Django a ten má své vlastní přihlašování a odhlašování, a proto není nutné řešit přihlašování a odhlašování.

Nutné je ovšem vytvoření uživatelského účtu, tedy registrace. Ta bude podrobněji rozebrána v následující kapitole.

6.5 Registrace

Uživatelský účet musí obsahovat dva základní povinné atributy, tedy uživatelské jméno a heslo.

Vzhledem k tomu, že může dojít ke ztrátě hesla, je nutné tuto situaci vyřešit.

Ve většině dnešních webových aplikací se tato situace řeší pomocí dalšího atributu a tedy přidáním emailu, na který je v případě ztráty zasláno nové náhodně vygenerované heslo, protože hesla jsou ukládána v zahashované podobě. Hashování je jednosměrný proces a tedy nelze získat hesla v původní podobě.

Framework Django má zabudovanou podporu uživatelských účtů. Ovšem je nutné nejprve uživatelský účet vytvořit,aby bylo možné jej poté využívat.

Proces vytvoření uživatelského účtu by měl obsahovat ověření emailu. Vzhledem k tomu, že proces implementace registrace uživatele s ověřením emailu je velmi opakovaným jevem, se kterým se musí programátoři webových aplikací potýkat a Django neumožňuje jednoduché řešení ověření emailu při registraci, vznikla aplikace django-registration.

6.5.1 Aplikace django-registration

`django-registration` [11] je aplikace, kterou vytvořil James Bennet a je šířena pod licencí, která umožňuje změny v kódu. Ty budou potřeba pro integraci a aplikací `pickswire`.

Aplikace umožňuje registraci uživatele v následujících krocích:

- Uživatel požádá o účet (provede zadání uživatelského jména, hesla a emailu)
- Uživatel na emailu obdrží instrukce k aktivaci účtu (tím dojde k ověření, že email opravdu existuje a uživatel nad ním má kontrolu a tedy v případě ztráty hesla lze na něj zaslat heslo nové)
- Uživatel aktivuje účet a může začít využívat výhod, které mu tento účet přináší oproti neregistrovaným uživatelům

Pro integraci `django-registration` je nutné ji nejprve nainstalovat.

Po stažení `django-registration` z oficiálních stránek je nutné přesunout složku do hlavního adresáře projektu, tedy do složky `grygar`.

Poté je nutné přidat do nastavení projetu k položce `INSTALLED_APPS` aplikaci `registration`. `django-registration` pro svou funkčnost vyžaduje instalaci vestavěných aplikací Django frameworku `django.contrib.auth` a `django.contrib.sites`. Tyto aplikace byly již popsány výše.

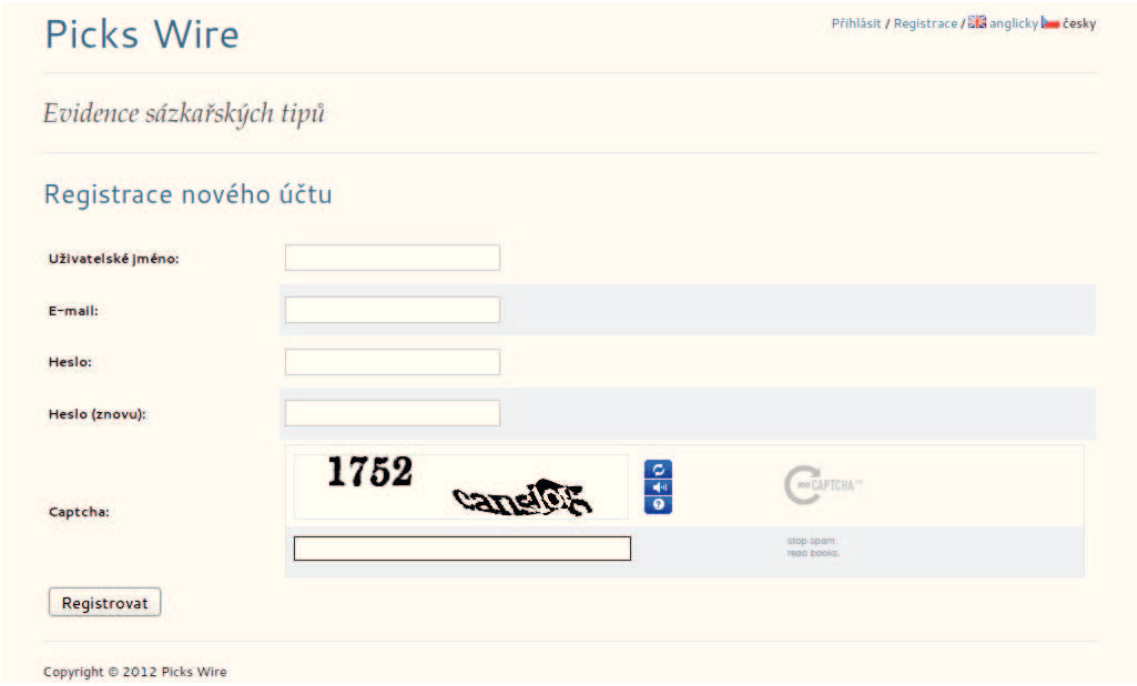
A dále je nutné nastavení pro emailové adresy, že které se bude email s aktivačními informacemi odesílat. Nastavení `ACCOUNT_ACTIVATION_DAYS` určuje kolik dní platí registrační odkaz zaslaný emailem. Další natavení se týkají přihlašovacích údajů k emailu a způsobu komunikace.

Poté ještě nutné aktivovat URL v souboru `url.py` projektu.

`django-registration` neobsahuje šablony ty je nutné vytvořit. Šablony neobsahují mnohé volně šiřitelné Django aplikace, ale `django-registration` obsahuje přehlednou dokumentaci pomocí které lze velmi jednoduše zjistit, které proměnné jsou šablonám předávány z pohledů.

Pro šablony je nutné vytvořit adresáře v aplikaci `registration` tyto adresáře budou pojmenovány dle konvencí Django frameworku. A také je nutné přidat do nastavení projektu cestu k adresářům.

Poté je tedy již možné začít vytvářet šablony. Základní šablona bude mít název `base.html` tento název je opět konvencí pojmenovávání šablon ve frameworku Django. Tato šablona bude pouze vyděděná z šablony `pickswire/base.html`, která slouží jako hlavní stránka projektu a obsahuje neměnné elementy (například hlavičku či patičku).



Obrázek 4: Registrační formulář s kontrolou CAPTCHA

Ostatní šablony jsou vyděděny z šablony `base.html` v aplikaci `registration`, pro potřeby aplikace `registration` vzniklo tedy celkem 8 šablon.

V neposlední řadě proběhla také úprava kódu, která nově registrovaného uživatele do skupiny s dostatečným oprávněním a nastavením přístupu do administračního rozhraní.

Registrační formulář je ovšem nutné ošetřit proti automatizovaným robotům, které by mohly automatizovaným vytvářením nesmyslných účtů záměrně přetěžovat aplikaci. A dále by se adresa, ze které bude email odesílán mohl dostat na černou listinu emailových schránek rozesílajících nevyžádanou poštu. Rovněž je potřeba chránit i formulář sloužící k obnovení hesla při jeho ztrátě.

K účelům ošetření formulářů aplikace `registration` jsem využil aplikaci `django-recaptcha`, která bude bude popsána v následující kapitole.

6.5.2 Aplikace django-recaptcha

CAPTCHA je Turingův test, který má za úkol odlišit uživatele od automatizovaných robotů.

Aplikace `django-recaptcha` [12] implementuje reCAPTCHA [13] od Google.

Pro užívání reCAPTCHA, je nutné se zaregistrovat a získat veřejný a privátní klíč pro danou doménu, tyto klíče se přidají do nastavení projektu a k položce `INSTALLED_APPS` přidat aplikaci `captcha`. Pro vývojáře je nutno dbát na to, že pokud pracuje na lokálním

serveru tak doména, pro kterou tyto klíče platí se liší od domény v produkčním nasazení, proto je nutné mít dvě sady klíčů.

Dále do nastavení projetu k položce `INSTALLED_APPS` přidat aplikaci `captcha` a vzhledem k tomu, že u `django-recaptcha` je dopředu uvažováno využití s `django-registration`, tak stačí nastavit URL v souboru `urls.py` projektu a řešení bude u registračního formuláře bude vždy vyžadována CAPTCHA.

Protože ověření pomocí CAPTCHA bude automaticky pouze u registračního formuláře je potřeba úprav kódu aplikace `django-registration`, aby byla CAPTCHA i u formuláře, který slouží v případě zapomenutého hesla, i tam totiž dochází k odesílání emailu. Jaké následky by mohlo mít neošetření takovéhoto formuláře pomocí CAPTCHA bylo popsáno v předchozí kapitole.

Jinak je potřeba zmínit, že užívání reCAPTCHA slouží k elektronizaci tištěných knih a relativně nedávno začal Google tento projekt využívat i pro čtení čísel domů. Takže užívání reCAPTCHA slouží pro dobrou věc.

Obrázek 4 zobrazuje registrační formulář, který obsahuje kontrolu CAPTCHA.

6.6 Internacionalizace a lokalizace

Internacionalizace je proces připravení softwaru na lokalizaci a provádí ji především vývojáři.

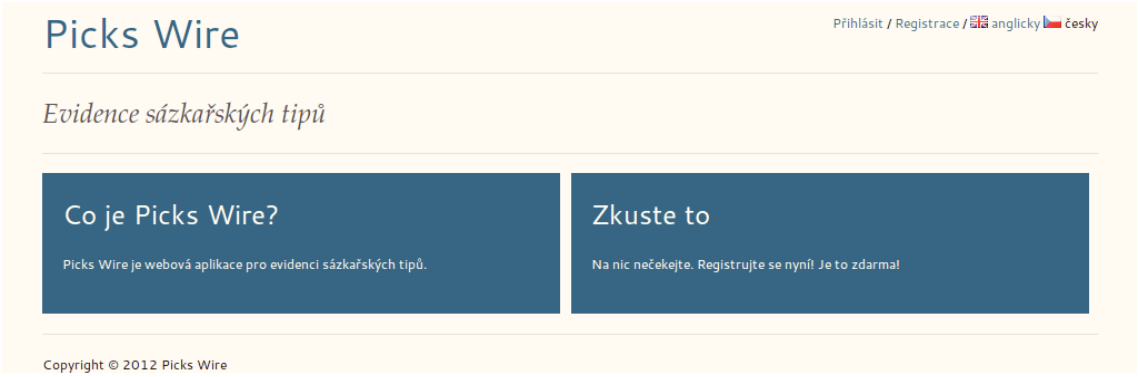
Lokalizace pak znamená překlad textových řetězců a je většinou jej provádí překladatel.

V nastavení je potřeba povolit internacionalizaci a lokalizaci. A zadat jazyky ve kterých bude aplikace dostupná.

V šablonách je potřeba označit řetězce k překladu, to samé je potřeba udělat také v šablonách. Jak již bylo zmíněno dříve celá aplikace je anglicky, tedy zdrojový kód má proměnné anglicky a řetězce k překladu jsou také anglicky, takže při lokalizaci stačí vytvořit překlad jen pro češtinu.

Nejprve je nutné vygenerovat soubory s překlady a ty poté přeložit. Překlady bude nutné provést pro aplikaci `pickswire` a `django-registration`. V adresáři projektu je potřeba vytvořit dle konvencí adresář `locale` a poté adresáři aplikace spustit příkaz `django-admin.py makemessages -l cs -e html,txt`, který vygeneruje soubory s řetězci k překladu. Poté je nutné tyto řetězce přeložit. Po překladu stačí spustit příkaz (opět v adresáři projektu) `django-admin.py compilemessages -l cs`, který převede soubory do binární podoby.

Jinak framework Django nově od verze 1.4 umožňuje mít v URL kód jazyka, což je dobré především v případě, že chceme aby byly stránky správně zaindexovány vyhledávači. Rozdílné jazykové mutace totiž musí mít unikání URL, protože se jedná o rozdílný obsah. Před verzí 1.4 bylo potřeba tuto funkčnost řešit rozhraním třetí strany.



Obrázek 5: Úvodní stránka

6.7 Úvodní stránka

Úvodní stránka má za cíl stručně a jasně informovat uživatele o jakou aplikaci se jedná a jaké výhody mu přináší a nalákat jej k registraci a následnému využívání služby. Anglicky se taková stránka označuje jako tzv. ”landing page”.

Kvůli úvodní stránce byla vytvořena nová šablona, že které dědí také šablony aplikace `django-registration`.

Pro grafické ztvárnění byl použit CSS framework Blueprint [14] a písma Google Web Fonts [15]. Finální podobu lze vidět na obrázku 5.

6.8 Inicializační data

V aplikaci bylo nutné vytvořit skupinu uživatelů aplikace `pickswire` a přiřadit ji patřičná práva, do této skupiny bude přiřazen nově registrovaný uživatel.

Dále byl v aplikaci vytvořen uživatel s uživatelským jménem `test` a heslem `test` a u tohoto uživatele byly vloženy ilustrační data, aby bylo možné vidět funkčnost aplikace bez nutnosti vytvářet nový účet a ten následně plnit daty.

Tato data byla exportována do souboru `initial_data.xml` v adresáři `fixtures` aplikace `pickswire`. Jedná se o inicializační data, tedy tato data budou do aplikace nahrána při vytváření.

Ale také je možné, data do databáze vkládat opakovaně. Například pokud nastane situace, že dáme uživatelům možnost vidět pomocí účtu `test` funkčnost aplikace a uživatel změní nějaká data, třeba všechny smaže, my ale budeme chtít aby funkčnost aplikace i s daty viděli i ostatní uživatelé a tak můžeme naplánovaným procesem, který bude reagovat například na odhlášení uživatele spustit opětovné nahrání dat do databáze.

7 Testování

Za účelem testování aplikace jsem oslovil malou skupinu sázkařů z komunitního webového fóra a na základě jejich připomínek jsem v aplikaci provedl úpravu aplikace tak aby tyto připomínky reflektovala.

Technické připomínky se týkaly například zaokrouhlování hodnot, kdy sázkaři upřesnili jaká přesnost by jim vyhovovala.

Dále profesionální sázkaři poukázali na fakt, že nevidí důvod proč by měli svá data o sázení poskytovat třetí osobě, protože nad daty nemají kontrolu a někdo by tak z jejich dat mohl odhadnout jejich taktiku sázení.

Dle mého osobního názoru touto paranoidní představou, že někdo takto zneužije data o sázení trpí především češi.

Řešením tohoto problému by mohlo být šifrování části dat uložených v databázi. V případě, že by byl databázový systém nějakým způsoben napaden a útočník by získal přístup k datům nebo celé databázi, nemohl by data zneužít. To ovšem není řešení pro sázkaře, kteří nechtějí dávat data třetí osobě, protože správce systému by měl klíč, kterým jsou data šifrována a správce tak vždy bude mít k datům přístup.

8 Nasazení na produkčním serveru

Pro nasazení jsem využil cloudové řešení Heroku.com, které umožňuje využití distribuovaného systému pro správu verzí s názvem Git.

8.1 Git

Git [16] je distribuovaný systém pro správu verzí, který vytvořil Linus Thordvalds za účelem vývoje jádra Linuxu.

Umožňuje spravovat jak velké tak i malé projekty. Mezi jeho hlavní přednosti patří jeho rychlost a možnosti nelineárního vývoje (vytváření a slučování větví).

8.2 Heroku.com

Heroku [17] je cloudová aplikační platforma, která funguje na principu PaaS (Platform as a service).

Cloud computing je model užívání a vývoje počítačových technologií. Jeho hlavní charakteristikou je to, že uživatelé platí především za užívání software, nikoliv za software samotný.

Hlavní výhody jsou vysoká škálovatelnost a spolehlivost. Vývojář se tak nemusí starat jak o hardware tak ani o správu softwaru, není tedy například nutné zaměstnávat administrátory serverů a starat se o správu serverů. Škálovatelnost je obrovskou výhodou u cloudových aplikací jak již bylo řečeno platíte za užívání služby. V případě, že začnou narůstat data v databázi nebo návštěvnost webové aplikace bude vyžadovat zvýšení prostředků cloudové řešení to svou elasticitou a škálovatelností hravě umožní.

Heroku nabízí účet, se kterým je možno vytvořit nekonečný počet aplikací, přičemž každá aplikace má 750 takzvaných dyno hodin měsíčně a 5 MB sdílené PostgreSQL databáze zdarma. Nevýhodou účtu, který je zdarma je, že pokud na aplikaci nepřijde po dobu jedné hodiny žádný požadavek proces se uspí a až při příchozím požadavku se probudí tato prodleva však trvá odhadem tak 5 sekund. Uspávání se totiž děje pouze pokud je k dispozici pouze jeden proces a u účtu zdarma je pouze jeden. Omezena je také celková velikost a také souborový systém neumožňuje zápis uživatelských dat.

8.3 Nasazení v praxi

Důležité je si uvědomit rozdíly v nastavení aplikace pro vývojové a produkční nasazení.

Nastavení projektu `grygar` se nachází v adresáři `settings`. V tomto adresáři jsou celkem tři soubory které se starají o nastavení. V souboru `base.py` veškeré nastavení, které je zároveň produkčním nastavením a v souboru `local.py` je nastavení pro vývoj, ale pouze to, které se od vývojového liší. Soubor `__init__.py` importuje soubor `base.py`, což je jak již bylo zmíněno soubor s veškerým produkčním nastavením a pokud existuje tak importuje i soubor `local.py`, tento soubor obsahuje pouze nastavení pro vývoj a nastavení v něm definovaná přepisují nastavení ze souboru `base.py`.

Nutností je tedy zajistit aby na produkční server nebyl nahrán soubor `local.py`. Vzhledem k tomu, že nasazení bude probíhat pomocí systému Git, vytvoří se v tomto adresáři také soubor `.gitignore`. V tomto souboru lze systému Git definovat, které soubory má ignorovat při nasazování aplikace na produkční server. V tomto případě v tomto souboru bude, že má ignorovat soubory `local.py` a `local.pyc`.

Jinak co se samotného nastavení týká tak, rozdílné bude například vypnutí lazení (anglicky debug) dále nastavení pro připojení k databázi, odkaz na URL pro statické soubory, které jsou při vývoji servírovány samotným serverem, ovšem v produkčním nastavení by to znamenalo značnou ztrátu výkonu. Při vývoji také využívám aplikaci `django-debug-toolbar.py`, která je při vývoji velmi užitečná, především při lazení databázových dotazů. A v neposlední řadě se liší privátní a veřejný klíč pro reCAPTCHA, protože jak již bylo zmíněno jedná se o rozdílné domény.

Hlavní změnou je však produkční server, při vývoji se pracuje s jednoduchým vývojovým serverem, který poskytuje samotný framework Django, ten se ovšem pro produkční nasazení nedoporučuje. Pro produkční nasazení byl tedy využit Green Unicorn [18], což je WSGI HTTP server napsaný v programovacím jazyce Python. V tomto případě ve verzi 14.2.

Do souboru `requirements.txt`, který se nachází v hlavním adresáři projektu je nutné vložit seznam modulů jazyka Python, které budou užity v produkčním nastavení. Framework Django ve verzi 1.4, adaptér pro databázi PostgreSQL psycopg2 a WSGI HTTP server Green Unicorn ve verzi 14.2.

Soubor `Procfile` v hlavním adresáři projektu obsahuje spouštěcí příkaz serveru Green Unicorn v produkčním nastavení.

V případě splnění výše uvedených nastavené je možné inicializovat nový projekt pomocí systému Git příkazem `git init`, poté přidat všechny soubory příkazem `git add .` a provést vložení první verze `git commit -m "initial commit"`.

Po instalaci klienta Heroku se vytvoří platforma jednoduše příkazem `heroku create pickswire --stack cedar`. A pomocí systému Git se provede nahrání aplikace do platformy příkazem `git push heroku master`.

Ještě je potřeba na produkčním serveru vytvořit databázi, to se na produkčním serveru provede příkazem `heroku run python manage.py syncdb`, nahrají se i inicializační data.

Aplikace je pak dostupná na doméně třetího řádu `http://pickswire.herokuapp.com`, ale samozřejmě Heroku podporuje i přidání vlastní domény.

8.4 Shrnutí

Aplikace je dostupná na `http://pickswire.herokuapp.com` a lze se přihlásit s uživatelským jménem `test` a heslem `test`, především v menu "Tikety" pak lze třídit podle mnoha kritérií včetně zobrazení zisku pomocí grafu.

K servírování statických souborů jsem v případě nasazení ukázky webu využil Google App Engine, protože na Heroku je aplikační platforma, to sice Google App Engine také, ale umožňuje lepší servírování statických souborů. Google App Engine se by ovšem

pouze pro servírování statických souborů používat nemělo, v opravdu ostrém nasazení bych využil Amazon Simple Storage Service (Amazon S3), který je k servírování statických souborů určen. Také v případě platformy Heroku je jasné, že při opravdu ostrém produkčním nasazení by bylo potřeba větší databáze minimálně dva procesy aby se nedocházelo k uspávání a další prostředky spojené s provozováním takovéto aplikace.

9 Obchodní model

Monetizace je proces, kdy převedeme neziskový web na ziskový web a tím začneme profitovat.

V případě této webové aplikace by monetizace mohla proběhnout formou affilate marketingu.

Affilate marketing je marketingový proces, kdy je poskytovatel reklamního prostoru ohodnocen pouze za proběhlé obchody.

Ovšem možnost monetizace pomocí affilate marketingu momentálně v České republice není možný, protože tento rok vstoupila v platnost novela loterijního zákona.

Tato novela říká že povoluje provozování loterijních a jiných her pouze právnickým osobám, které mají sídlo na území České republiky a také zakazuje reklamu na subjekty, které toto povolení nemají.

To fakticky znamená odstřihnutí zahraničních sázkových sázkových kanceláří a proto je možnost affilate marketingu, který by fungoval tak, že na webu by byly reklamy sázkových kanceláří, které by lákaly uživatele k registraci k určité sázkové kanceláři a po registraci a splnění podmínek affilate programu by byla vyplacena provize nemožná. Protože odstřihnutí zahraničních sázkových kanceláří znamená značné zmenšení trhu.

Monetizace formou zobrazování reklamy a platbou za proklik, tzv. PPC, také není ve většině případů možné, protože weby spjaté s kurzovým sázením porušují pravidla těchto programů.

10 Závěr

V této práci jsem nastínil problém, se kterým se setkávají sportovní sázkaři, tedy důležitost vedení evidence výsledku, která může dopomoci ke zlepšení výsledků v budoucnosti.

Navrhl jsem a implementoval webovou aplikaci, která tento problém řeší.

Během implementace jsem odhalil slabá místa frameworku Django, například že signál se spustí pouze při uložení jednotlivé instance, nikoliv při uložení několika instancí naráz. Při vazbě M:N se vazba uloží až po signálu `post_save` dané instance a tak nelze v metodě tohoto signálu pracovat s takto nově vytvořenou či aktualizovanou vazbou, ale je nutné řešit tuto situaci jinak, například odchycením dat z formuláře jak tomu bylo v tomto případě.

Provedl jsem nasazení aplikace v cloudové platformě, která z hlediska budoucího rozvoje aplikace umožní zvládat případný nápor uživatelů, díky škálovatelnosti a elasticitě tohoto řešení.

Dalšímu rozvoji aplikace by mohla pomoci komunita sázkařů, na jejichž zpětné vazbě by bylo možné části aplikace vylepšovat dle jejich potřeb. A v neposlední, řadě by monetizaci aplikace mohla dopomoci změna zákonů v České republice, která by otevřela trh zahraničním sázkovým kancelářím.

11 Reference

[1] Pilgrim, M., *Dive into Python* Apress, 2004.

[2] Schmuller, J., *Teaching Yourself UML in 24 Hours* Sams, 1999.

[3] Holovaty A., Kaplan–Moss J., *The Definitive Guide To Django: Web Development Done Right 2nd Edition* Apress, 2009.

[4] *jQuery: The Write Less, Do More, JavaScript Library* [online]. 2012 [cit. 2012-04-29]. Dostupné z: http://jquery.com/

[5] *Python Programming Language – Official Website* [online]. 1990 [cit. 2012-04-29]. Dostupné z: http://python.org/

[6] *Django — The Web framework for perfectionists with deadlines* [online]. 2005 [cit. 2012-04-29]. Dostupné z: https://www.djangoproject.com/

[7] *PostgreSQL: Welcome* [online]. 1996 [cit. 2012-04-29]. Dostupné z: http://www.postgresql.org/

[8] *Google Chart Tools — Google Developers* [online]. 2012 [cit. 2012-04-29]. Dostupné z: https://developers.google.com/chart/

[9] *danawoodman/jquery-showhide · GitHub* [online]. 2012 [cit. 2012-04-29]. Dostupné z: https://github.com/danawoodman/jquery-showhide

[10] *carhartl/jquery-cookie · GitHub* [online]. 2012 [cit. 2012-04-29]. Dostupné z: https://github.com/carhartl/jquery-cookie

[11] *ubernostrum / django-registration / overview — Bitbucket* [online]. 2012 [cit. 2012-04-29]. Dostupné z: https://bitbucket.org/ubernostrum/django-registration/

[12] *praekelt/django-recaptcha · GitHub* [online]. 2012 [cit. 2012-04-29]. Dostupné z: https://github.com/praekelt/django-recaptcha

[13] *reCAPTCHA: Stop Spam, Read Books* [online]. 2012 [cit. 2012-04-29]. Dostupné z: http://www.google.com/recaptcha

[14] *Blueprint: A CSS Framework — Spend your time innovating, not replicating* [online]. 2012 [cit. 2012-04-29]. Dostupné z: http://blueprintcss.org/

[15] *Google Web Fonts* [online]. 2012 [cit. 2012-04-29]. Dostupné z: https://www.google.com/webfonts

[16] *Git - Fast Version Control System* [online]. 2012 [cit. 2012-04-29]. Dostupné z: http://git-scm.com/

[17] *Heroku — Cloud Application Platform* [online]. 2012 [cit. 2012-04-29]. Dostupné z: http://www.heroku.com/

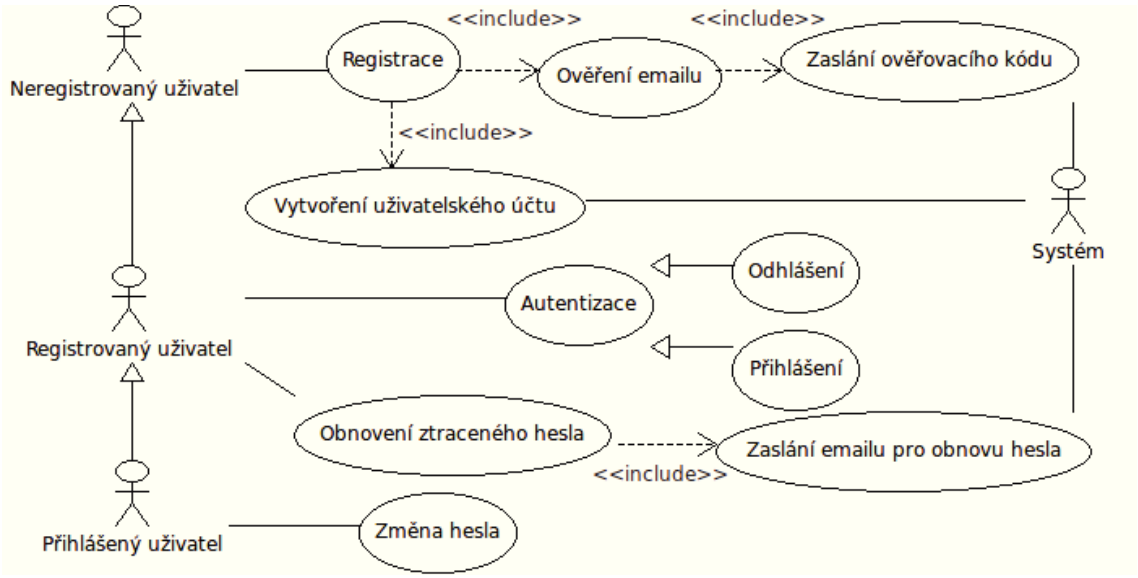
[18] *Green Unicorn - Welcome* [online]. 2012 [cit. 2012-04-29]. Dostupné z: http://gunicorn.org/

[19] *Sports Betting Record Keeping - Betmagnet.com* [online]. 2011 [cit. 2012-04-29]. Dostupné z: http://www.betmagnet.com/

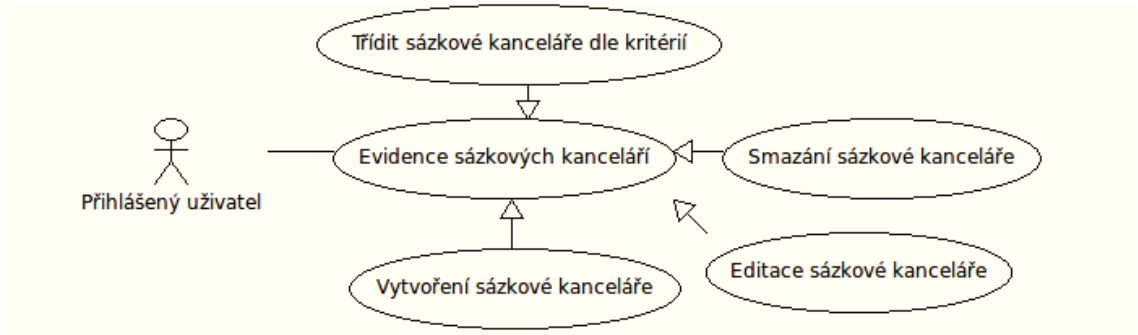
[20] *Betbook - Sázkařská komunita* [online]. 2012 [cit. 2012-04-29]. Dostupné z: http://www.betbook.com/

A Diagramy případů užití

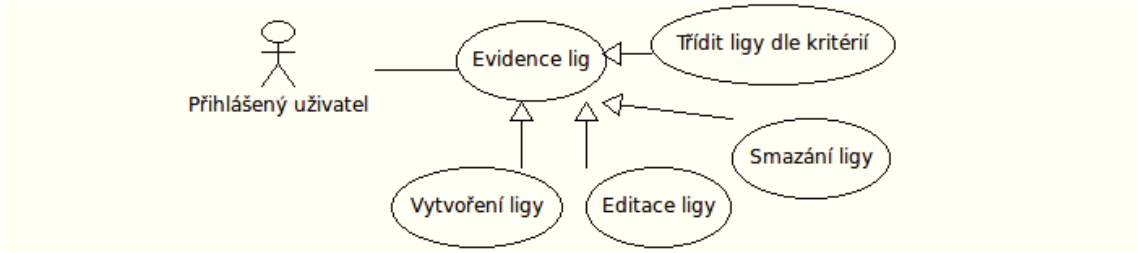
Obrázek 6: Aktéři



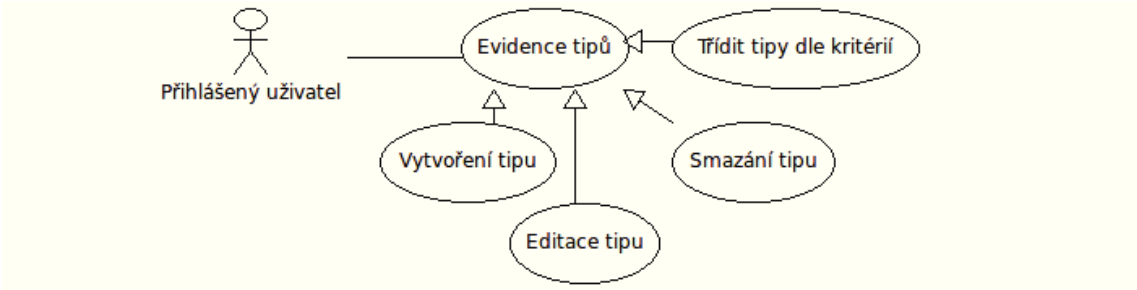
Obrázek 7: Práce s uživatelským účtem



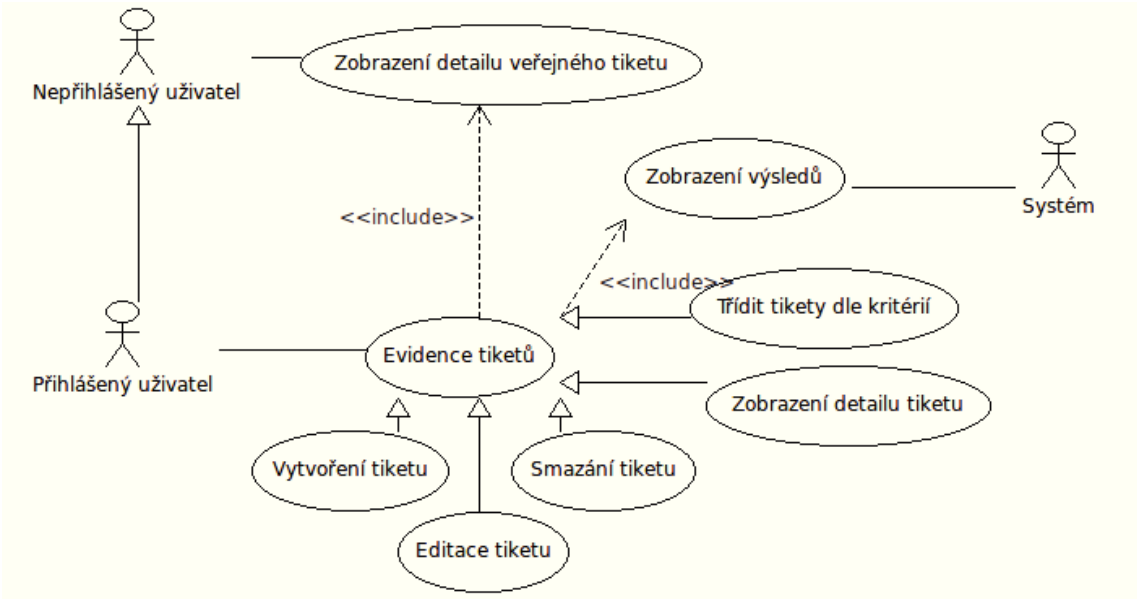
Obrázek 8: Evidence sázkovách kanceláří



Obrázek 9: Evidence lig



Obrázek 10: Evidence tipů



Obrázek 11: Evidence tiketů

B Úplné tabulky atributů

Název atributu	Datový typ	Klíč	NULL	Index	Integritní omezení
id	INTEGER	PK	ne	ano	
name	VARCHAR(200)	ne	ne	ano	
currency	VARCHAR(3)	ne	ne	ano	výběr možností
url	VARCHAR(200)	ne	ne	ne	platný formát URL
author_id	INTEGER	FK	ne	ano	

Tabulka 2: Tabulka Bookmaker

Název atributu	Datový typ	Klíč	NULL	Index	Integritní omezení
id	INTEGER	PK	ne	ano	
sport	VARCHAR(100)	ne	ne	ano	výběr možností
origin	VARCHAR(20)	ne	ne	ne	výběr možností
shortcut	VARCHAR(10)	ne	ne	ano	
name	VARCHAR(200)	ne	ne	ano	
official_site_url	VARCHAR(200)	ne	ne	ne	platný formát URL
author_id	INTEGER	FK	ne	ano	

Tabulka 3: Tabulka League

Název atributu	Datový typ	Klíč	NULL	Index	Integritní omezení
id	INTEGER	PK	ne	ano	
event_date	TIMESTAMP	ne	ne	ano	
event	VARCHAR(200)	ne	ne	ne	
pick	VARCHAR(200)	ne	ne	ne	
pick_type	SMALLINT	ne	ne	ne	výběr možností
odd	NUMERIC(8,3)	ne	ne	ne	min. hodnota 1,001
notes	CLOB	ne	ne	ne	
status	SMALLINT	ne	ne	ne	výběr možností
result	VARCHAR(200)	ne	ne	ne	
league_id	INTEGER	FK	ne	ano	

Tabulka 4: Tabulka Pick

Název atributu	Datový typ	Klíč	NULL	Index	Integritní omezení
id	INTEGER	PK	ne	ano	
stake	NUMERIC(17,2)	ne	ne	ne	
handling_fee	DOUBLE	ne	ne	ne	výběr možností
trust	DOUBLE	ne	ne	ne	výběr možností
is_public	BOOLEAN	ne	ne	ne	
creation_date	TIMESTAMP	ne	ne	ne	
picks_count	SMALLINT	ne	ne	ne	
paid	NUMERIC(17,2)	ne	ne	ne	
odd	NUMERIC(8,3)	ne	ne	ne	
edoldp*	TIMESTAMP	ne	ne	ne	
status	SMALLINT	ne	ne	ne	
net_profit	NUMERIC(17,2)	ne	ne	ne	
bookmaker_id	INTEGER	FK	ne	ano	

Tabulka 5: Tabulka Ticket

Název atributu	Datový typ	Klíč	NULL	Index	Integritní omezení
id	INTEGER	PK	ne	ano	
ticket_id	INTEGER	FK	ne	ano	
pick_id	INTEGER	FK	ne	ano	

Tabulka 6: Tabulka Ticket_picks

Název atributu	Datový typ	Klíč	NULL	Index	Integritní omezení
id	INTEGER	PK	ne	ano	
username	VARCHAR(30)	ne	ne	ne	
first_name	VARCHAR(30)	ne	ne	ne	
last_name	VARCHAR(30)	ne	ne	ne	
email	VARCHAR(70)	ne	ne	ne	
password	VARCHAR(128)	ne	ne	ne	
is_staff	BOOLEAN	ne	ne	ne	
is_active	BOOLEAN	ne	ne	ne	
is_superuser	BOOLEAN	ne	ne	ne	
last_login	TIMESTAMP	ne	ne	ne	
date_joined	TIMESTAMP	ne	ne	ne	

Tabulka 7: Tabulka User

C Příloha na DVD

Obsah DVD:

```
+-- grygar-lukas-gry147-bp-2012.pdf (bakalářská práce ve formátu PDF)
+-- source-code (adresář se zdrojovými kódy)
    +-- grygar
        +-- captcha (adresář aplikace třetí strany)
            +-- ..
        +-- grygar (hlavní adresář projektu)
            +-- django_1_4_admin_static
                +-- ..
            +-- settings
                +-- ..
            +-- static
                +-- ..
            +-- __init__.py
            +-- url.py
            +-- wsgi.py
        +-- LICENSES
        +-- pickswire (adresář aplikace pro evidenci tipů)
            +-- fixtures
                +-- initial_data.xml
            +-- locale
                +-- ..
            +-- static
                +-- ..
            +-- templates
                +-- ..
            +-- admin.py
            +-- forms.py
            +-- __init__.py
            +-- list_filters.py
            +-- models.py
            +-- signals.py
            +-- urls.py
            +-- views.py
        +-- registration (adresář aplikace třetí strany)
            +-- ..
        +-- manage.py
        +-- Procfile
        +-- requirements.txt
```